Color processing

Dr. Su Su Maung
CEIT Department
Yangon Technological University

Outline of Lecture8

- Introduction
- Perceptual aspects of color
- Color models
- Color images in Python
- Processing of color images

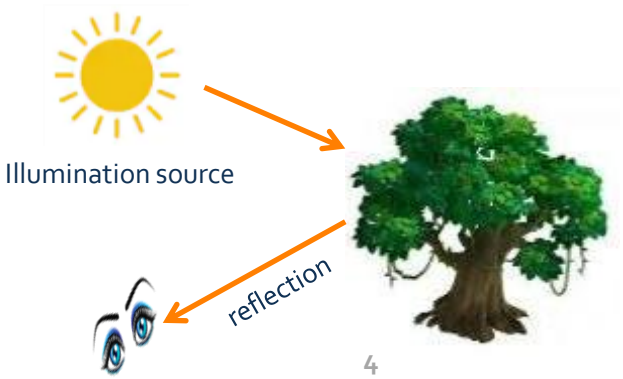
Introduction

- For human beings, Color is a powerful descriptor that simplifies object identification and extraction from a scene. The human visual system is particularly attuned to two things: edges, and color.
- We have mentioned that the human visual system is not particularly good at recognizing subtle changes in grey values.
- In this section we shall investigate color briefly, and then some methods of processing color images.

What is color?

- Color study consists of
  - the physical properties of light which give rise to color,
  - the nature of the human eye and the ways in which it detects color,
  - the nature of the human vision centre in the brain, and the ways in which messages from the eye are perceived as color.

The color that human perceive in an object = the light reflected from the object



Perceptual aspects of color

- The human visual system tends to perceive color as being made up of varying amounts of red, green and blue. That is, human vision is particularly sensitive to these colors; These values are called the primary colors.

- If we add together any two primary colors we obtain the secondary colors:

magenta (purple) = red + blue
cyan = green + blue
yellow = red + green

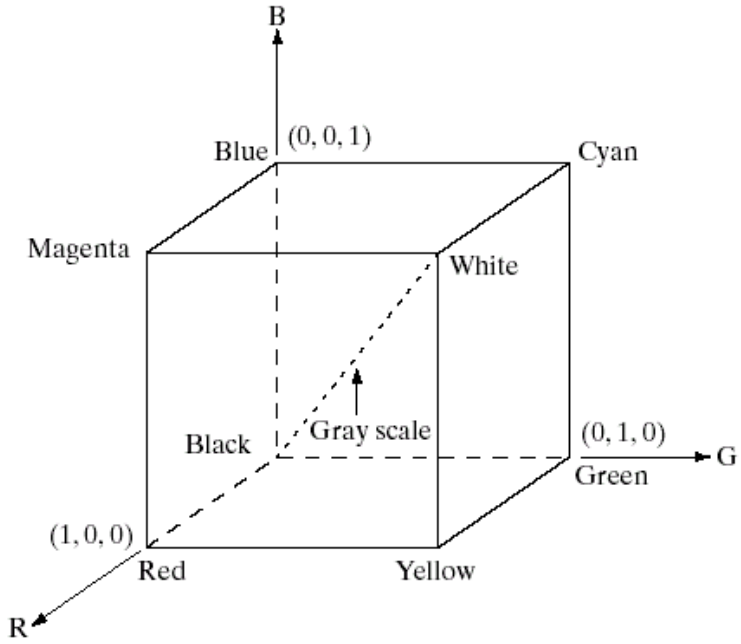
Color models

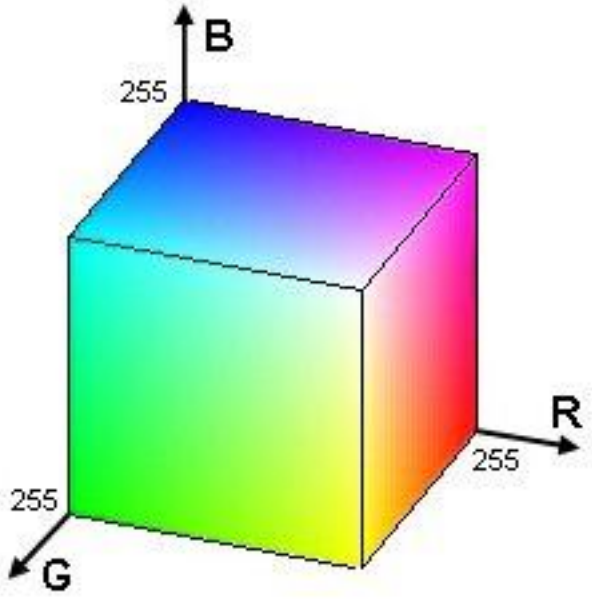
- A color model is a method for specifying colors in some standard way.
- We shall investigate three systems.
  - RGB
  - HSV
  - YIQ

The RGB color model

- RGB is a color model based on additive primary colors
- Red—Specifies the intensity of red as an integer between 0 and 255. A color with red set to 0 specifies the absence of color and emits no red light. A color with red set to 255 will appear bright red, or fully saturated with color.
- Green—Specifies the intensity of green as an integer between 0 and 255. A color with green set to 0 specifies the absence of color and emits no green light. A color with green set to 255 will appear bright green, or fully saturated with color.
- Blue—Specifies the intensity of blue as an integer between 0 and 255. A color with blue set to 0 specifies the absence of color and emits no blue light. A color with blue set to 255 will appear bright blue, or fully saturated with color.

The RGB color model





$$\left(2^8\right)^3 = 16{,}777{,}216 \text{ Colors}$$

True color image

Ture color image

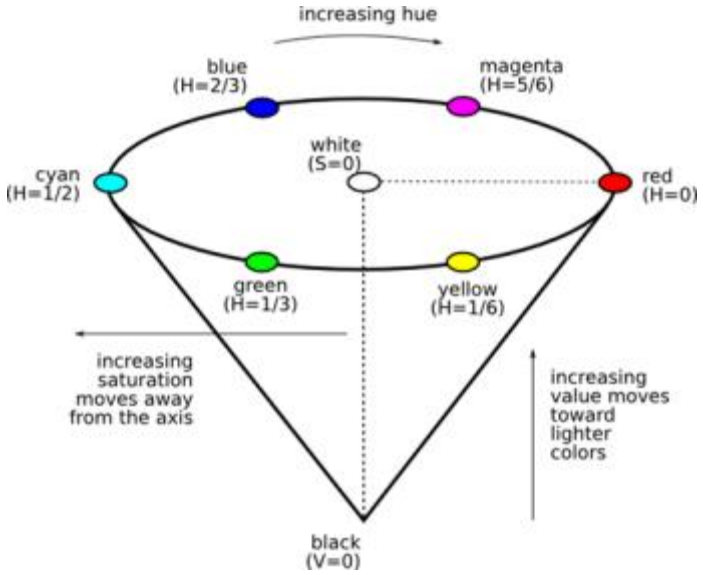
Red

Green

Blue

The HSV color model

- The HSV color model is based on a color system in which the color space is represented by a single cone. The three components of the cone are the hue, saturation, and value.



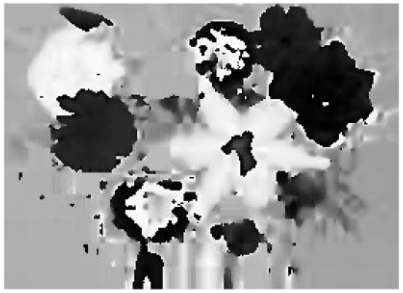
Color	Hue
Red	0
Yellow	0.1667
Green	0.3333
Cyan	0.5
Blue	0.6667
Magenta	0.8333

HSV

- HSV stands for Hue, Saturation, Value. These terms have the following meanings:
  - Hue: The true color attribute (red, green, blue, orange, yellow, and so on).
- Saturation: The amount by which the color has been diluted with white.
  - The more white in the color, the lower the saturation. So a deep red has high saturation, and a light red (a pinkish color) has low saturation.
  - if one or two of the RGB values are zero, the saturation will be one, the highest possible value.
- Value: The degree of brightness(intensity): a well lit color has high intensity; a dark color has low intensity.

HSV (cont.)

Color image



Hue

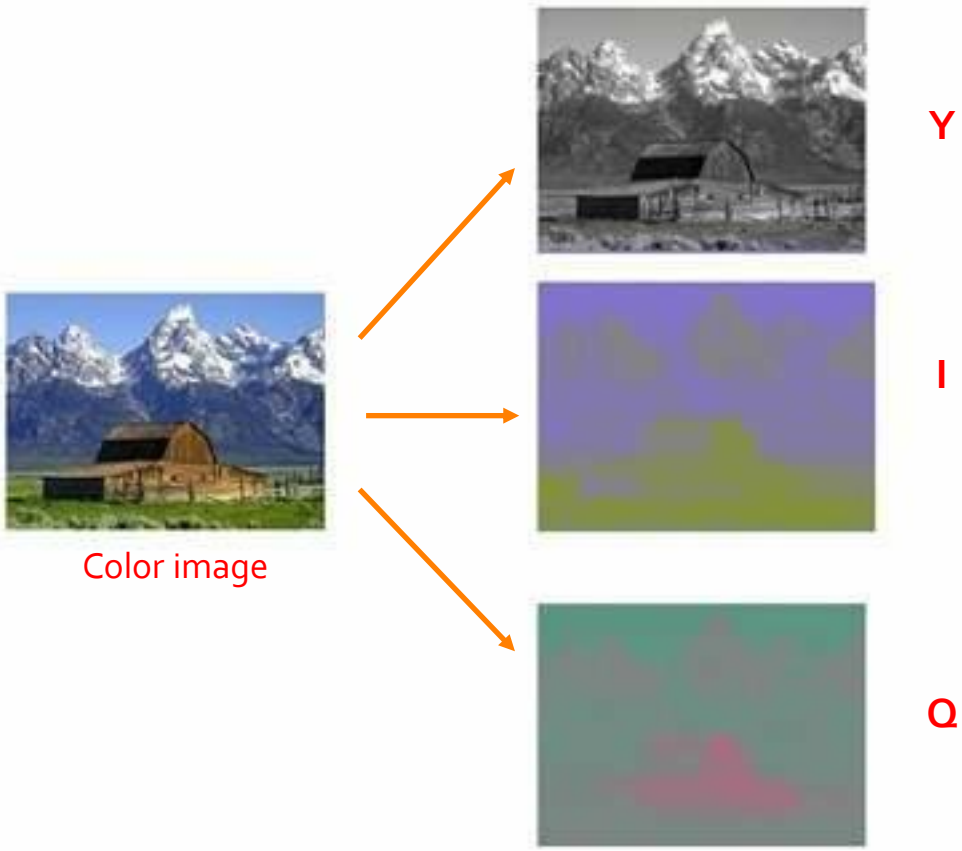
Saturation

Value

YIQ

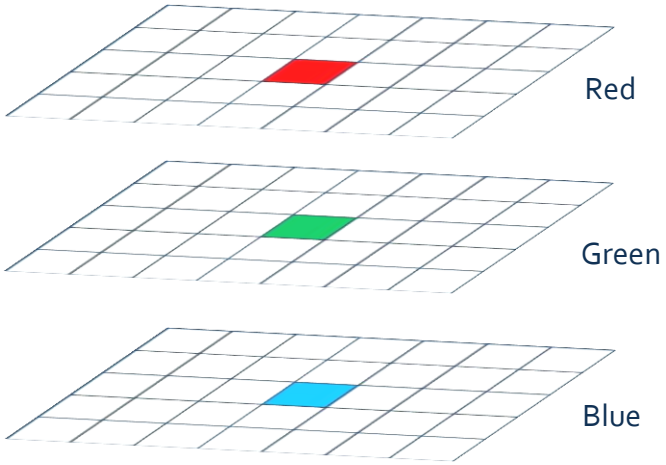
- This color space is used for TV/video in America and other countries where NTSC(**National Television System Committee**) is the video standard (Australia uses PAL(Phase Alternating Line)).
- One of the main advantages of this format is that gray-scale information is separate from color data, so the same signal can be used for both color and monochrome television sets.
- In this scheme Y is the luminance (this corresponds roughly with intensity), and hue (I) and saturation (Q).
- The luminance component represents gray-scale information, and the other two components carry the color information of a TV signal.

YIQ (cont.)



Color images in Python

- Since a color image requires three separate items of information for each pixel, a (true) color image of size m x n is represented in Python by an array of size m x n x 3 : a three dimensional array



A three dimensional array for an RGB image

Color images in Python (cont.)

```
img=io.imread(images/tulip.jpg')
img.shape
Out[66]: (225, 225, 3)
```

- We can isolate each color component by the colon operator:
  - img[:,:,1] The first, or red component
  - img[:,:,2] The second, or green component
  - img[:,:,3] The third, or blue component
- These can all be viewed with io.imshow():

```
for i in range(3):
   plt.figure();io.imshow(img[:,:,i])
   io.imsave('img'+np.str(i)+'.jpg',img[:,:,i],)
```

Color images in Python (cont.)

A color image

Red component

Green component

Blue component

An RGB color image and its components

Colour images in Python (cont.)

- Conversion to HSV

```
import skimage.color as co
img=io.imread('images/flower.jpg')
io.imshow(img)
imghsv=co.rgb2hsv(img)
for i in range (3):
    plt.fugure(i); io.imshow(imghsv[:,:,i]
```

A color image

Hue

Saturation

Value

The HSV components

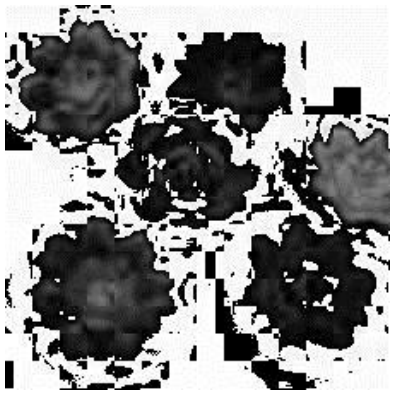
Colour images in Python

- Conversion to YIQ

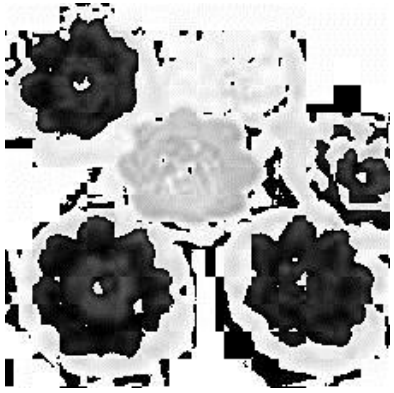
```
import colorsys as color
img=io.imread('images/waterlily7.jpg')
imgyiq=color.rgb_to_yiq(*np.dsplit(img,3))
for i in range(3):
    plt.figure(i);io.imshow(imgyiq[i][:,:,0])
```

A color image

Y



I



Q

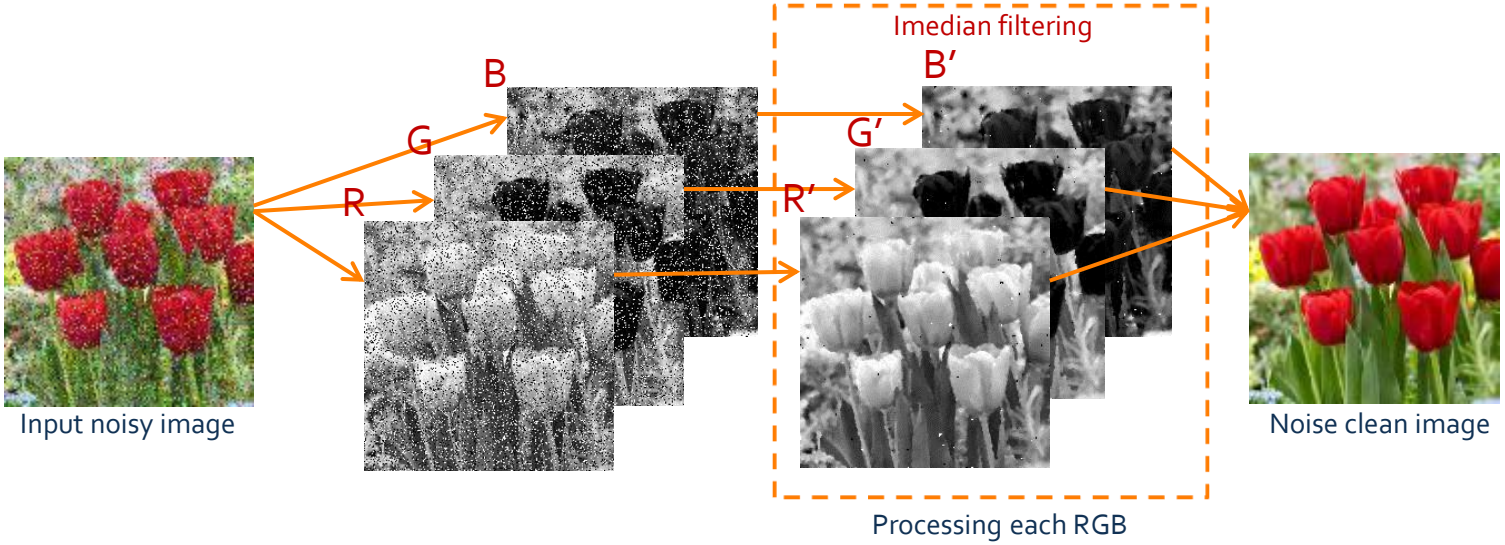
The YIQ components

Notice that the Y component of YIQ gives a better greyscale.

Processing of color images

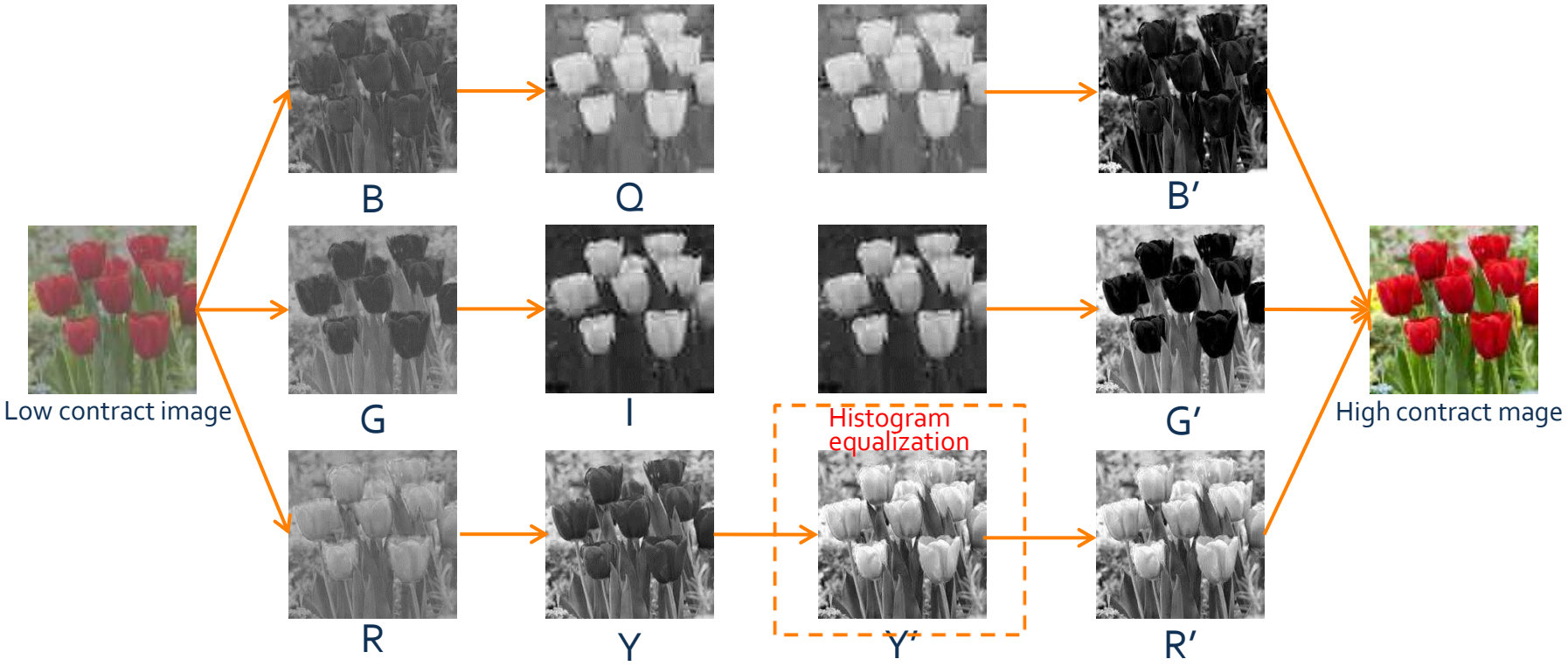
- There are two methods we can use:
  - process each R, G, B matrix separately,
  - transform the color space to one in which the intensity is separated from the color, and process the intensity component only.
- We shall apply either of the above schema to color images for a number of different image processing tasks.

RGB processing



RGB processing

Intensity processing



- Convert from RGB to YIQ.
- Process the Y component only

Intensity processing

Contrast Enhancement

- one way of enhancing the contrast in a color image would be to enhance each of the color layers separately. Each layer can be processed separately, and the final result viewed.
- This better result is done by processing the intensity component. Now we have to convert from RGB to YIQ, apply histogram equalization to the intensity component, and convert back to RGB for display:
- we try to apply histogram equalization to each of the RGB components: some strange colors have been introduced.

Contrast Enhancement (cont.)

Using each RGB component

Intensity processing

Histogram equalization of a color image

Noise reduction

- The method of noise removal must depend on the generation of noise. In the above example we assumed that the noise was generated after the image had been acquired and stored as RGB components.
- But as noise can arise anywhere in the image acquisition process, it is quite reasonable to assume that noise might affect only the brightness of the image.
- In such a case denoising the Y component of YIQ will produce the best results.

Noise reduction (cont.)

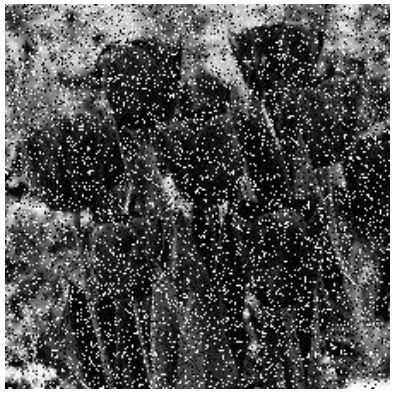
- We would apply median filtering to each of the RGB components and the intensity component only.
- We see that we can't apply the median filter to the intensity component only, because the conversion from RGB to YIQ spreads the noise across all the YIQ components. If we remove the noise from Y only, then the noise has been only slightly diminished as shown in Figure.
- If the noise applies to only one of the RGB components, then it would be appropriate to apply a denoising technique to this component only.

Noise reduction (cont.)

Salt & pepper noise

The red component

The green component



The blue component

Noise on a color image

Noise reduction (cont.)

```
import skimage.io as io
import numpy as np
import scipy.ndimage as ndi
imgn=io.imread('images/noise_tulip.jpg')
img[:,:,0]=ndi.median_filter(imgn[:,:,0],3)
img[:,:,1]=ndi.median_filter(imgn[:,:,1],3)
img[:,:,2]=ndi.median_filter(imgn[:,:,2],3)
```

```
import skimage.io as io
import skimage.color as co
import scipy.ndimage as ndi
imgn=io.imread('noise_tulip.jpg')
imgh=co.rgb2hsv(imgn)
imgh[:,:,2]=ndi.median_filter(imgh[:,:,2],5)
img=co.hsv2rgb(imgh)
```

Denoising each RGB component

Denoising Y only

Attempts at denoising a color image

Edge detection

- An edge image will be a binary image containing the edges of the input. We can go about obtaining an edge image in two ways:
  - we can take the intensity component only, and find its edges,
  - we can find the edges of each of the RGB components, and join the results

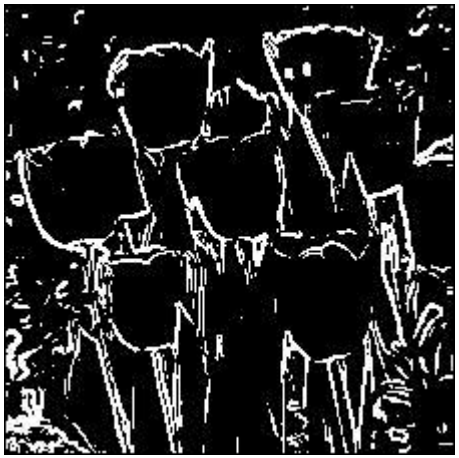
Edge detection (cont.)

- To implement the first method, we start with the rgb2gray function:
- For the second method, the results of the Sobel filteringwill be cast to numeric values.
- The edge image is a much more complete edge image.

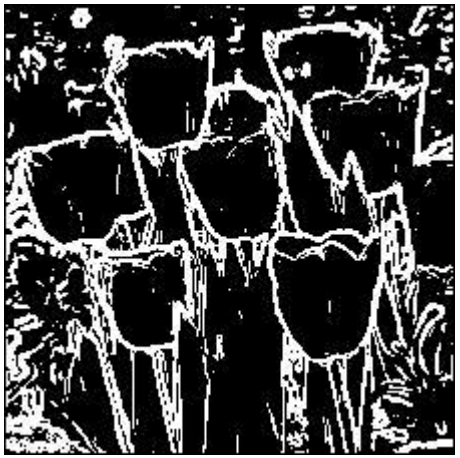
```
import skimage.io as io
import skimage.color as co
import skimage.filters as fl
img = io.imread('tulip.jpg')
imgg = co.rgb2gray(img)
imggf = fl.sobel(imgg)>0.12
io.imsave('edge_gray.jpg',np.uint8(imggf*255))
```

```
import skimage.io as io
import skimage.filters as fl
img = io.imread('tulip.jpg')
img2 = np.zeros_like(img)
for i in range(3):
   img2[:,:,i]=fl.sobel(img[:,:,i])>0.12
imge2 = (img2[:,:,0]+img2[:,:,1]+img2[:,:,2])>0
```

Edge detection (cont.)



Edges of a gray image



Edges of each RGB component

The edges of a color image

Next Week Lecture (Week9)

- Lecture9:Image coding and compression

Thank You